LabRandomForest.R

karanYsingh

2021-04-26

```
# Chapter 8 Lab: Decision Trees

# Fitting Classification Trees

library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.5
```

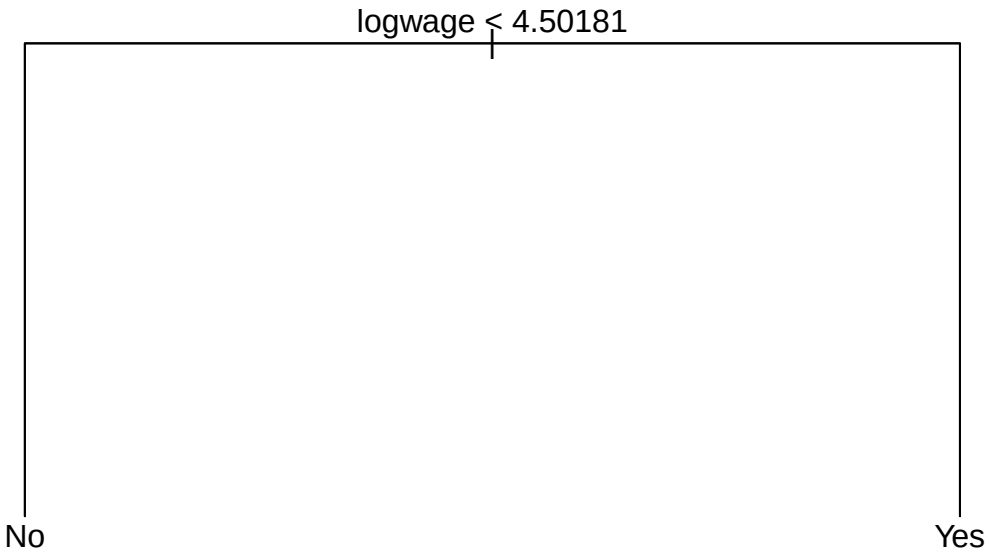
```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.4
```

```
attach(Wage)
High=factor(ifelse(wage<=90,"No","Yes"))
Wage=data.frame(Wage,High)
tree.wage=tree(High~.-wage,Wage)
summary(tree.wage)
```

```
##
## Classification tree:
## tree(formula = High ~ . - wage, data = Wage)
## Variables actually used in tree construction:
## [1] "logwage"
## Number of terminal nodes:  2
## Residual mean deviance:  0 = 0 / 2998
## Misclassification error rate: 0 = 0 / 3000
```

```
plot(tree.wage)
text(tree.wage,pretty=0)
```




```
tree.wage
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
## 1) root 3000 3723 Yes ( 0.3117 0.6883 )
##   2) logwage < 4.50181 935    0 No ( 1.0000 0.0000 ) *
##   3) logwage > 4.50181 2065    0 Yes ( 0.0000 1.0000 ) *
```

```
set.seed(2)
train=sample(1:nrow(Wage), 200)
Wage.test=Wage[-train,]
High.test=High[-train]
tree.wage=tree(High~.-wage,Wage,subset=train)
tree.pred=predict(tree.wage,Wage.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred   No  Yes
##       No   859    0
##       Yes    0 1941
```

```
(83+117)/200
```

```
## [1] 1
```

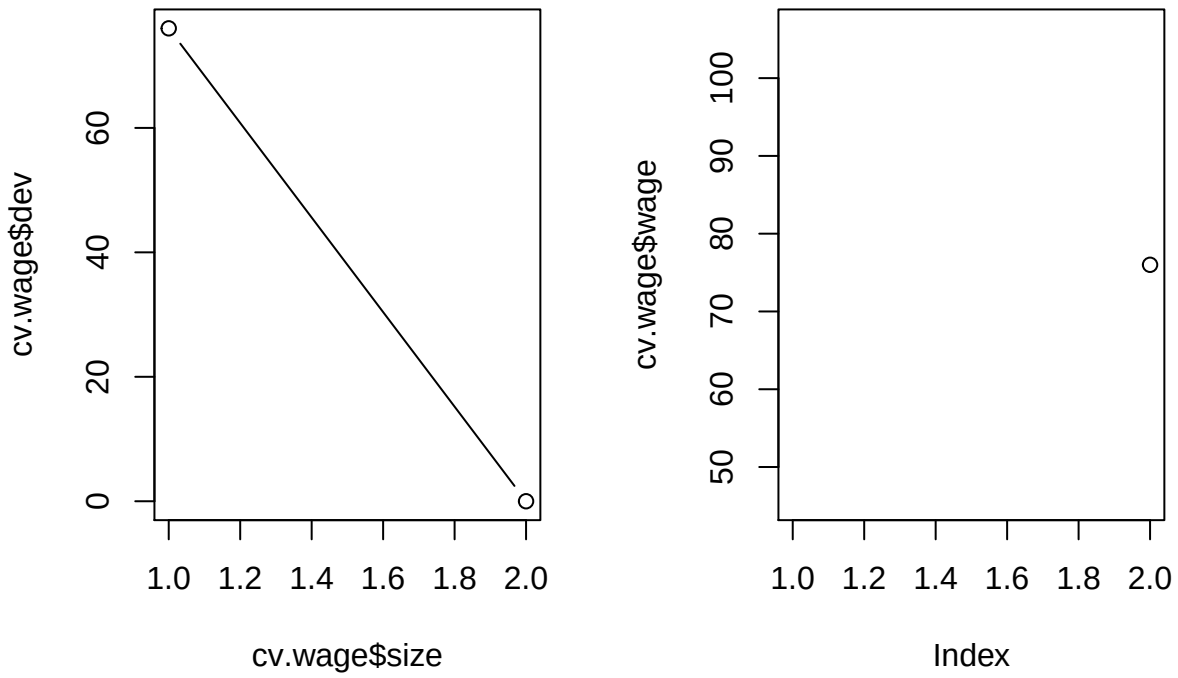
```r
set.seed(3)
cv.wage=cv.tree(tree.wage,FUN=prune.misclass)
names(cv.wage)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
cv.wage
```

```
## $size
## [1] 2 1
##
## $dev
## [1]  0 76
##
## $k
## [1] -Inf   76
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
par(mfrow=c(1,2))
plot(cv.wage$size,cv.wage$dev,type="b")
plot(cv.wage$k,cv.wage$wage,type="b")
```




```
prune.wage=prune.misclass(tree.wage,best=9)
```

```
## Warning in prune.tree(tree = tree.wage, best = 9, method = "misclass"): best is
## bigger than tree size
```

```
plot(prune.wage)
text(prune.wage,pretty=0)
tree.pred=predict(prune.wage,Wage.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred   No  Yes
##       No   859    0
##       Yes    0 1941
```

```
(859+1941)/(859+1941)
```

```
## [1] 1
```

```
prune.wage=prune.misclass(tree.wage,best=15)
```

```
## Warning in prune.tree(tree = tree.wage, best = 15, method = "misclass"): best is
## bigger than tree size
```

```
plot(prune.wage)
text(prune.wage,pretty=0)
```


```
tree.pred=predict(prune.wage,Wage.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred   No  Yes
##       No   859    0
##       Yes    0 1941
```

```
(86+62)/200
```

```
## [1] 0.74
```

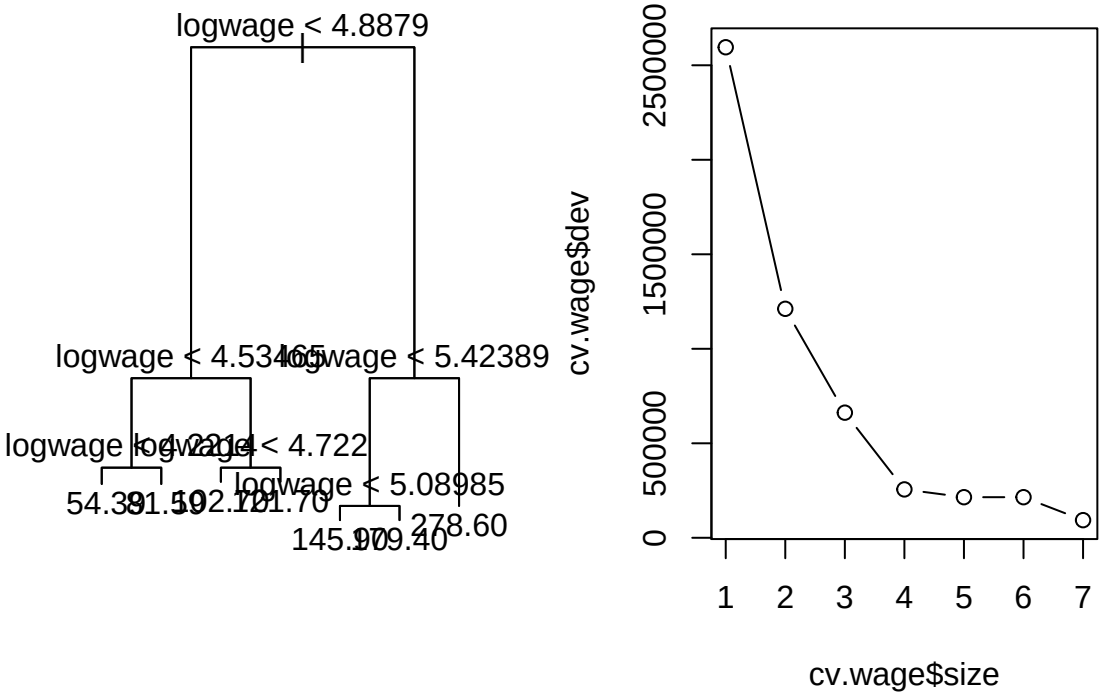
```
# Fitting Regression Trees

set.seed(1)
train = sample(1:nrow(Wage), nrow(Wage)/2)
tree.wage=tree(wage~.,Wage,subset=train)
summary(tree.wage)
```

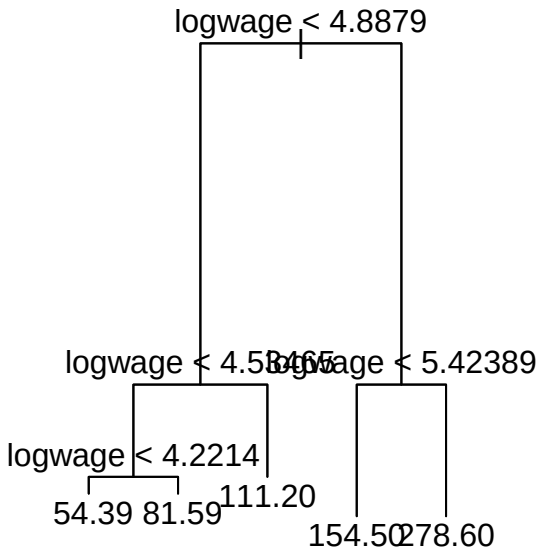
```
##
```

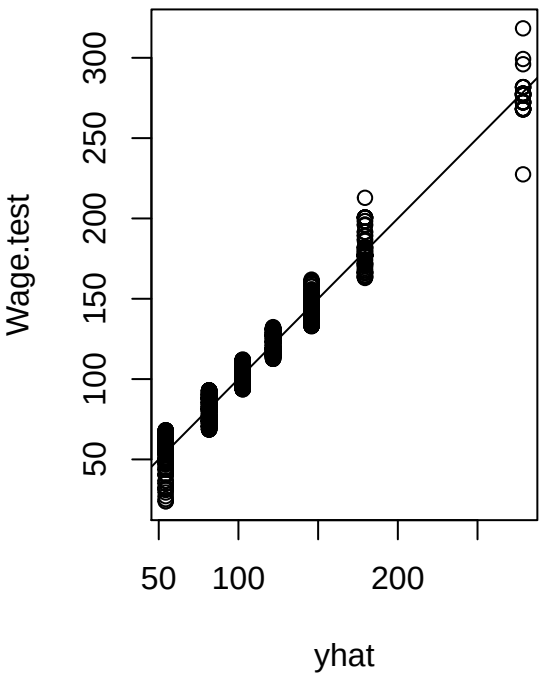
```
## Regression tree:
## tree(formula = wage ~ ., data = Wage, subset = train)
## Variables actually used in tree construction:
## [1] "logwage"
## Number of terminal nodes:  7
## Residual mean deviance:  60.12 = 89750 / 1493
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -34.3000  -5.2510  -0.3052   0.0000   6.3930  39.7700
```

```
plot(tree.wage)
text(tree.wage,pretty=0)
cv.wage=cv.tree(tree.wage)
plot(cv.wage$size,cv.wage$dev,type='b')
```




```
prune.wage=prune.tree(tree.wage,best=5)
plot(prune.wage)
text(prune.wage,pretty=0)
yhat=predict(tree.wage,newdata=Wage[-train,])
Wage.test=Wage[-train,"wage"]
plot(yhat,Wage.test)
abline(0,1)
```






```
mean((yhat-Wage.test)^2)
```

```
## [1] 59.45303
```

```
# Bagging and Random Forests

library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.wage=randomForest(wage~.,data=Wage,subset=train,mtry=13,importance=TRUE)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
bag.wage
```

```
##
## Call:
##  randomForest(formula = wage ~ ., data = Wage, mtry = 13, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##           Mean of squared residuals: 0.3986436
##                     % Var explained: 99.98
```

```
yhat.bag = predict(bag.wage,newdata=Wage[-train,])
plot(yhat.bag, Wage.test)
abline(0,1)
mean((yhat.bag-Wage.test)^2)
```

```
## [1] 0.6150102
```

```
bag.wage=randomForest(wage~.,data=Wage,subset=train,mtry=13,ntree=25)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
yhat.bag = predict(bag.wage,newdata=Wage[-train,])
mean((yhat.bag-Wage.test)^2)
```

```
## [1] 0.7007993
```

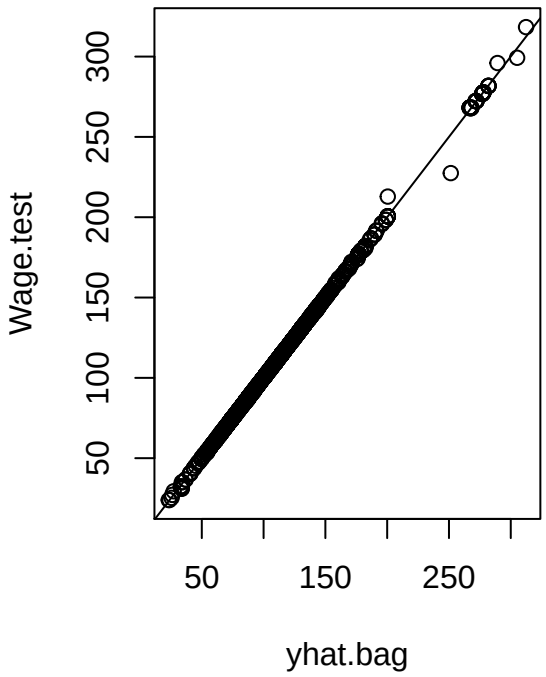
```
set.seed(1)
rf.wage=randomForest(wage~.,data=Wage,subset=train,mtry=6,importance=TRUE)
yhat.rf = predict(rf.wage,newdata=Wage[-train,])
mean((yhat.rf-Wage.test)^2)
```

```
## [1] 2.499548
```

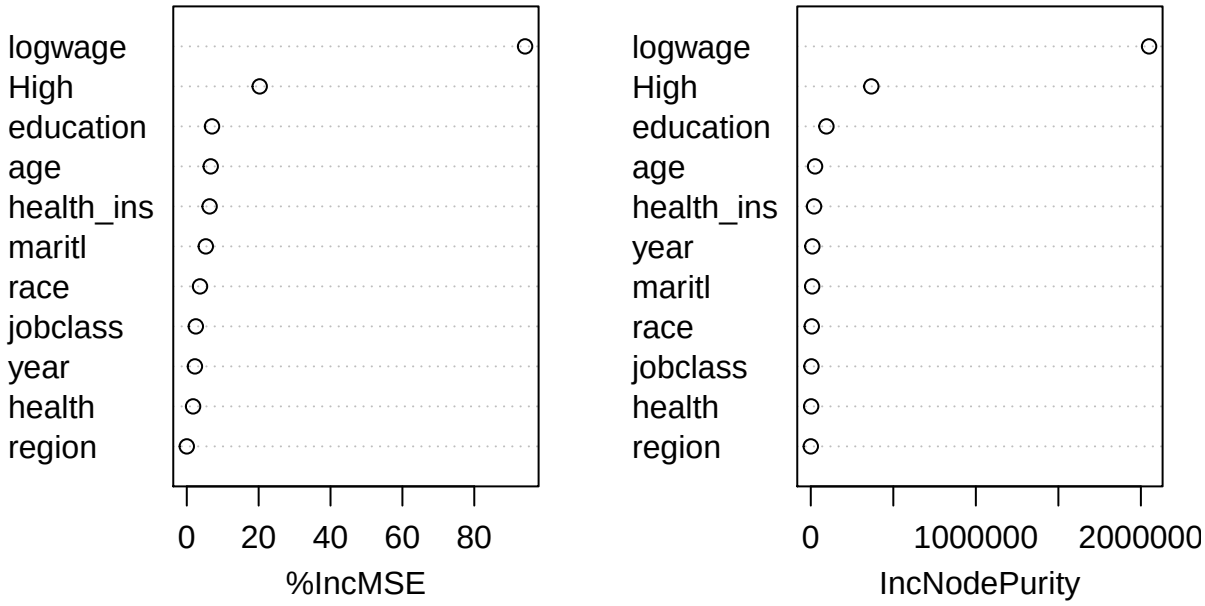
```
importance(rf.wage)
```

```
##              %IncMSE IncNodePurity
## year        2.262629      9533.877
## age         6.657540     25911.870
## maritl      5.292742      8390.332
## race        3.678778      5853.513
## education   7.018106     94647.878
## region      0.000000         0.000
## jobclass    2.518688      3853.744
## health      1.752768      2210.533
## health_ins  6.354622     19661.615
## logwage    94.135829   2049459.919
## High       20.289262    366711.041
```

```
varImpPlot(rf.wage)
```



rf.wage




```
oob.err = double(13)
test.err = double(13)
for(mtry in 1:13){
  fit=randomForest(wage~.,data=Wage,subset = train,mtry=mtry,ntree=400)
  oob.err[mtry]=fit$mse[400]
  pred=predict(fit,Wage[-train,])
  test.err[mtry]=with(Wage[-train,],mean((wage-pred)^2))
  cat(mtry," ")
}
```

```
## 1  2  3  4  5  6  7  8  9  10  11
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
## 12
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
## 13
```

```
matplot(1:mtry,cbind(test.err,oob.err),pch=19,col=c('green','blue'),type="b",ylab="Mean Squared Error")
legend("topright",legend=c("OOB","Test"),pch=19,col=c("red","blue"))
```